МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДНІПРОВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. О. ГОНЧАРА

КАФЕДРА МАТЕМАТИЧНОГО ЗАБЕЗПЕЧЕННЯ ЕОМ

МОДУЛЬНА РОБОТА

з дисципліни «Програмування та алгоритмічні мови»

Виконав: студент групи ПС-18-1

Гаєвський Дмитро

Перевірив:

Дніпро, 2019

# Зміст

[Зміст 2](#_Toc10122124)

[Постановка задачі 3](#_Toc10122125)

[Теоретична частина 3](#_Toc10122126)

[Використання структурного типу 3](#_Toc10122127)

[Робота з файлами 3](#_Toc10122128)

[Найбільш використовувані операції при роботі із файлами[2] 4](#_Toc10122129)

[Алгоритмічна частина 6](#_Toc10122130)

[Опис програми та даних 7](#_Toc10122131)

[Передача у програму шляху до вхідного файлу 7](#_Toc10122132)

[Структура введення даних 7](#_Toc10122133)

[Опис введених типів 7](#_Toc10122134)

[Передача у програму шляху до вихідного файлу 7](#_Toc10122135)

[Структура даних вихідного файлу 7](#_Toc10122136)

[Інтерфейс користувача 7](#_Toc10122137)

[Тестування 8](#_Toc10122138)

[Вхідний файл: 8](#_Toc10122139)

[Вихідний файл: 8](#_Toc10122140)

[Консольний вивід: 9](#_Toc10122141)

[Аналіз результатів 9](#_Toc10122142)

[Джерела 10](#_Toc10122143)

[Додаток 11](#_Toc10122144)

# Постановка задачі

Скласти програму, яка вводить дані про оцінки абітурієнтів по 8 задачах для кожного, а також вектор балів за кожне завдання. Підрахувати для кожного абітурієнта загальну суму балів з урахуванням того, що оцінки нижче 3 (у 5-ти бальній системі) не враховуються. По кожному завданню порахувати відсоток абітурієнтів, що його не розв'язали. Сформулювати масив номерів абітурієнтів, сума балів у яких перевищує задане “порогове” значення. Надрукувати початкові дані і результати в наочному вигляді, з позначеннями.

# Теоретична частина

## Використання структурного типу

Структура — це сукупність різнотипних елементів, яким присвоюється одне ім’я. В останніх версіях С++ до складу структури можна включати не тільки поля, але і методи. Можливість перевизначати оператори, у тому числі і операторів введення/виведення (>>, <<), та public рівень доступу членів робить структурі дуже зручними для використання при зчитуванні/записі даних у файл.

## Робота з файлами

Стандартна бібліотека мови С ++ містить визначення типу istream для потоків введення і типу ostream - для потоків виведення.

Потік ostream робить наступне.

* Перетворює значення різних типів в послідовності символів.
* Посилає ці символи "кудись" (наприклад, на консоль, у файл, основну пам'ять або на інший комп'ютер).

Потік istream робить наступне.

* Перетворює послідовності символів в значення різних типів.
* Отримує ці символи "звідкись" (наприклад, з консолі, з файлу, з основної пам'яті або від іншого комп'ютера).

Якщо ви хочете зчитати дані з файлу або записати їх у файл, відкрийте потік конкретно для цього файлу, потік ifstream - це потік istream для читання з файлу, потік ofstream - це потік ostream для запису в файл, а потік fstream є потік iostream, який можна використовувати як для читання, так і для запису. Перед використанням файловий потік повинен бути пов'язаний з файлом, наприклад:

ifstream ist{iname } ; // ist - вхідний потік для файлу iname

if ( ! ist) error ( " Неможливо відкрити вхідний файл ", name);

ofstream ost {оnаше}; // os t - вихідний потік для файлу oname

if (! ost) еrrоr ("Неможливо відкрити вихідний файл ", оnаmе);

Визначення потоку ifstream з ім'ям, заданим рядком name, відкриває файл з цим ім'ям для читання. Перевірка !ist дозволяє з'ясувати, чи був файл відкритий коректно. Після цього можна зчитувати дані з файлу точно так же. як з будь-якого іншого потоку istream. Аналогічно для ofstream. Наприклад, якщо припустити, що оператори введення (>>) та виведення (<<) визначені для типу Player. Тоді можна написати наступний фрагмент програми:

Player p;

ist>>p;

ost<<p;

Коли файловий потік виходить з області видимості, пов'язаний з ним файл закривається. Коли файл закривається, пов'язаний з ним буфер скидається; інакше кажучи, символи з буфера записуються в файл[1].

### Найбільш використовувані операції при роботі із файлами[2]

#### Відкриття

Відкрити файл можна двома способами: викликавши метод open() або вказавши шлях до нього в конструкторі. Вам необхідно підготувати текстовий файл, перед тим, як почати набирати код. Створіть файл з розширенням txt - "файл.txt".

Використання методу open() зручно, якщо програміст не хоче відразу прив'язуватися до файлу. Якщо ж потрібно відкрити файл всередині якоїсь функції, попрацювати з ним і закрити, то можна прописати шлях до файлу прямо в конструкторі.

Відкривши файл, бажано перевірити: чи відкрився він, оскільки є ряд причин, за якими файл може не відкритися, а ми цього не побачимо. Наприклад, файлу з вказаним ім'ям немає в прописаної папці або шлях вказано невірно. Для цього використовується метод is\_open (): метод поверне 1, якщо файл знайдений і успішно відкритий, інакше поверне 0.

Якщо файл успішно відкритий, з нього можна зчитувати.

#### Оператор зчитування >>

Так само як і в iostream зчитування можна організувати оператором >>, який вказує в яку змінну буде вироблено зчитування. Варто зазначити, що оператор >> застосовується до текстових файлів. Зчитування з бінарного файлу виробляти найкраще за допомогою методу read ().

До речі цей оператор досить зручний, якщо стоїть завдання розділити файл на слова.

#### Методи getline () і get ()

Зчитування цілого рядка до перекладу каретки проводиться так само як і в iostream методом getline().

Принцип в загальному той же, що і в аналогах з iostream: Вказується в параметрах буфер (змінна, куди буде проводитися читання), або точніше покажчик на блок пам'яті (якщо змінна оголошена статично: char buffer [255] Наприклад, то пишеться в параметри &buffer), вказується максимальна кількість яку зчитує, щоб не сталося переповнення і вихід за межі буфера і в разі потреби символ-роздільник, до якого буде зчитування.

#### Метод read ()

Відміна read() в тому, що цей метод застосовується для неформатованого введення.

#### Метод close ()

Закриває файл. Це потрібно робити, по-перше, для економії пам’яті, по-друге, стаються рідкісні ситуації, коли відкритий для читання файл псується, якщо завершити програму не закриваючи файл.

#### Метод eof ()

Перевіряє чи досягнений кінець файла. Тобто чи можна з нього продовжувати читання. Вище приклад з зчитування слів оператором >> якраз використовує таку перевірку.

# Алгоритмічна частина

Підключення бібліотек

Опис структури student:

Поля:

Телефонний номер - number

Ім’я - firstName

Фамілія – LastName

Бали - grade

Функція main()

Оголошення локальних змінних

Відкриття потоків введення/виведення

Якщо потоки були відкриті успішно:

Зчитування вектора балів за кожну задачу

Доки не досягли кінця файлу:

Зчитування даних про наступного абітурієнта

Перетворення оцінок в бали

Підрахунок загальної кількості балів

Якщо загальна кількість балів перевищує прохідний поріг:

Зберегти дані абітурієнта

Вивести загальну кількість балів абітурієнта

Кінець циклу

Вивести данні про студентів, бали яких перевищили прохідний поріг

Цикл по задачам:

Підрахувати відсоток абітурієнтів, які не впорались із задачею

Вивести цей відсоток

Вивести дані по кожній задачі

Кінець mein()

# Опис програми та даних

## Передача у програму шляху до вхідного файлу

* Шлях до вхідного файлу задається одразу у вихідному коді програми.
* Вхідний файл повинен знаходитись в одній директорії з .exe файлом програми та мати назву “in.txt”

## Структура введення даних

* Перший рядок файлу означає вектор коефіцієнтів для кожної задачі
* Усі подальші рядки означають абітурієнтів. Спочатку йде номер абітурієнту, потім ім'я, прізвище та оцінки по кожній задачі

## Опис введених типів

* В першому рядку повинно бути 8 коефіцієнтів, по одному на кожну задачу
* У кожному наступному рядку:
  + Номер абітурієнта повинен буди до 13 символів включно
  + Ім’я та прізвище до 20 символів включно кожне
  + 8 оцінок за кожне завдання по 5-бальній системі

## Передача у програму шляху до вихідного файлу

* Шлях до вихідного файлу задається одразу у вихідному коді програми.
* Вхідний файл буде знаходитись в одній директорії з .exe файлом програми та мати назву “out.txt”

## Структура даних вихідного файлу

* Спочатку записується ім’я, прізвище та загальна кількість балів для кожного абітурієнта
* Далі після рядку з «Student numbers with a passing grade:» записані рядки з номерами та ініціалами студентів, що пройшли поріг «склав»
* Далі йдуть 8 рядків з відсотком абітурієнтів, які не зробили задачу, та її номер, наприклад: «Percentage of not solving the task #4: 100»
* Далі йдуть 8 рядків векторів балів по кожній задачі

## Інтерфейс користувача

* Програма має консольний інтерфейс
* Користувачеві потрібно ввести прохідний бал
* Структура виводу програми до консолі абсолютно аналогічна до структури виводу у вихідний файл (див. Структура даних вихідного файлу)

# Тестування

## Вхідний файл:

1 1.2 0.9 0.25 0.5 0.75 0.8 0.9

+380985409283 Dima Haievskyi 5 5 4 5 4 5 5 5

+380989127389 Nikita Ivanov 3 2 4 5 2 3 1 3

+380982345678 Petiy Sidorov 1 3 4 5 2 4 5 5

## Вихідний файл:

Dima Haievskyi 30.1

Nikita Ivanov 12.8

Petiy Sidorov 19.95

Student numbers with a passing grade:

+380985409283 Dima Haievskyi

+380982345678 Petiy Sidorov

Percentage of not solving the task #1: 33.3333

Percentage of not solving the task #2: 33.3333

Percentage of not solving the task #3: 0

Percentage of not solving the task #4: 0

Percentage of not solving the task #5: 66.6667

Percentage of not solving the task #6: 0

Percentage of not solving the task #7: 33.3333

Percentage of not solving the task #8: 0

Vector of task #1: 5 3 1

Vector of task #2: 6 2.4 3.6

Vector of task #3: 3.6 3.6 3.6

Vector of task #4: 1.25 1.25 1.25

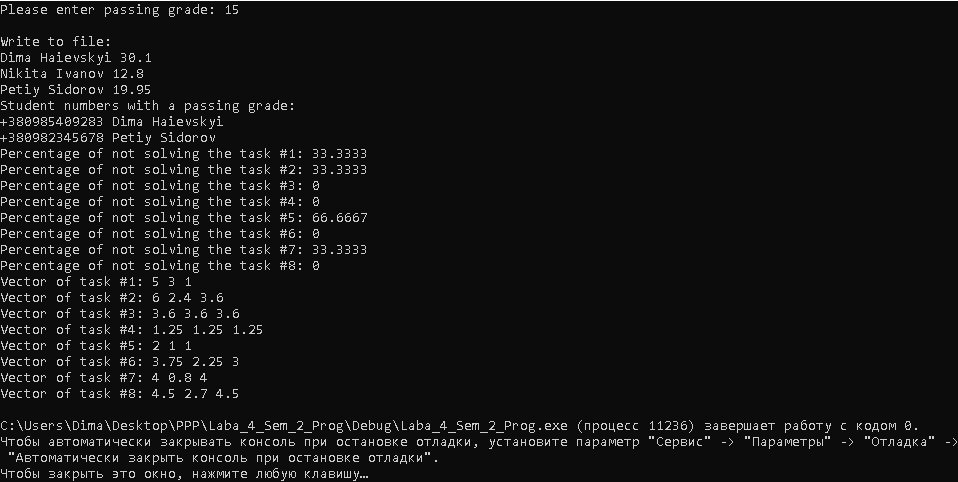
Vector of task #5: 2 1 1

Vector of task #6: 3.75 2.25 3

Vector of task #7: 4 0.8 4

Vector of task #8: 4.5 2.7 4.5

## Консольний вивід:



## Аналіз результатів

Підрахувавши кількість білів для кожного абітурієнта за допомогою коефіцієнтів в початку файла маємо:

Dima Haievsly: 5 + 6 + 3.6 + 1.25 + 2.5 + 3.75 + 4 + 4.5 = 30.1

Nikita Ivanov: 3 + 2.4 + 3.6 + 1.25 + 1 + 2.25 + 0.8 + 2.7 = 12.8

Petiy Sidorov: 1 + 3.6 + 3.6 + 1.25 + 1 + 3 + 4 + 4.5 = 19.95

Тобто пройшли поріг «склав» 2 абітурієнти, перший та третій, йх данні були виведенні. Далі можна з легкістю перевірити відсоток тих, хто не впорався з завданням: наприклад в 5 задачі абітурієнти мають такі бали: 4, 2, 2, а отже другий та третій не впоралися з завданням (набрали менше 3 балів), тобто ~66% не впоралися (2/3)

Далі виводиться вектор балів для кожної задачі. Він повністю співпадає з балами абітурієнтів.

Отримані розрахунки повністю відповідають виводу програми.

# Джерела

1. Страуструп, Бьярне. *Программирование: принципы и практика с использованием С++, : Пер. с англ. - М. : ООО "И.Д. Вильяме", 2016. - 1 328 с. : ил. - Парал. тит. англ*
2. [purecodecpp.com](https://purecodecpp.com/archives/2751)

# Додаток

#include "pch.h"

#include <iostream>

#include <fstream>

#include <string.h>

#include <vector>

using namespace std;

struct student {

char number[58];

char firstName[20];

char lastName[20];

double grade[8];

};

struct g {

char number[58];

};

int main() {

int i, j;

double coef[8], score, passing;

vector <double> v[8];

vector <g> good;

g g1;

student s;

ifstream fin("in.txt");

ofstream fout("out.txt");

cout << "Please enter passing grade: ";

cin >> passing;

if (!fin)

cout << "Unable to open file in.txt";

else if (!fout)

cout << "Unable to open file out.txt";

else {

cout << "\nWrite to file:\n";

for (i = 0; i < 8; i++)

fin >> coef[i];

while (!fin.eof()) {

score = 0;

fin >> s.number;

fin >> s.firstName;

fin >> s.lastName;

for (i = 0; i < 8; i++) {

fin >> s.grade[i];

v[i].push\_back(s.grade[i] \* coef[i]);

if (s.grade[i] > 2)

score += s.grade[i] \* coef[i];

}

if (score > passing) {

strcpy\_s(g1.number, s.number);

strcat\_s(g1.number, " ");

strcat\_s(g1.number, s.firstName);

strcat\_s(g1.number, " ");

strcat\_s(g1.number, s.lastName);

good.push\_back(g1);

}

fout << s.firstName << " " << s.lastName << " " << score << "\n";

cout << s.firstName << " " << s.lastName << " " << score << "\n";

}

fout << "Student numbers with a passing grade:\n";

cout << "Student numbers with a passing grade:\n";

for (i = 0; i < good.size(); i++) {

fout << good[i].number << "\n";

cout << good[i].number << "\n";

}

for (i = 0; i < 8; i++) {

score = 0;

for (j = 0; j < v[0].size(); j++)

if (v[i][j] / coef[i] < 3)

score++;

fout << "Percentage of not solving the task #" << i + 1 << ": " << score \* 100.0 / v[0].size() << "\n";

cout << "Percentage of not solving the task #" << i + 1 << ": " << score \* 100.0 / v[0].size() << "\n";

}

for (i = 0; i < 8; i++) {

fout << "Vector of task #" << i + 1 << ": ";

cout << "Vector of task #" << i + 1 << ": ";

for (j = 0; j < v[0].size(); j++) {

fout << v[i][j] << " ";

cout << v[i][j] << " ";

}

fout << "\n";

cout << "\n";

}

fin.close();

fout.close();

}

}